VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2013 David Beinhauer

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Interaktivní výukový kurz – Microsoft Silverlight a Windows Phone 7

Interactive tutorial – Microsoft Silverlight and Windows Phone 7

2013 David Beinhauer

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **David Beinhauer**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Interaktivní výukový kurz - Microsoft Silverlight a Windows Phone 7
Interactive tutorial - Microsoft Silverlight and Windows Phone 7

Zásady pro vypracování:

Cílem této práce je vytvořit výukový kurz pokrývající vývoj aplikací pro platformu Windows Phone 7 pomocí technologie MS Silverlight.

1. Navrhněte strukturu a obsah výukového kurzu pokrývajícího využití technologie MS Silverlight pro platformu Windows Phone 7.
2. Vytvořte obsah kurzu a výukový kurz implementujte. Výukový kurz bude realizován formou internetově přístupného obsahu.
3. Samotný kurz doplňte a rozšiřte o výukové prvky ve formě multimédií, screencastu, příkladů, úkolů, apod.
4. V rámci obsahu kurzu odlište úrovně zkušeností uživatele/studenta a podle tohoto jej navigujte napříč vysvětlovanými tématy a informacemi.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Radecký, Ph.D.**

Datum zadání: 18.11.2011
Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Poděkování

Předně bych chtěl poděkovat svému vedoucímu bakalářská práce, kterým je Ing. Michal Radecký, Ph.D., za možnost výběru tohoto tématu a také za podnětnou diskuzi v průběhu tvorby práce.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 7. 5. 2013

.....

David Beinhauer

Abstrakt

Cílem bakalářské práce je seznámit čtenáře s vývojem aplikací pro mobilní operační systém Windows Phone 7. Bakalářská práce prezentuje základní aspekty vývoje pro tuto platformu za pomoci technologie Silverlight a vychází z přiložené příručky, která je součástí práce ve formě přílohy, jakožto i online verze této příručky, která se nachází na adrese http://www.winphone.cz.

Klíčová slova

Silverlight, Windows Phone, XAML, C#, vývoj, mobilní aplikace, Microsoft, Mango, kurz

Abstract

Bachelor thesis is an introduction to development for mobile operating system Windows Phone 7. It describes the basics how to develop for this platform using the Silverlight Framework and it is based on the manual which is a part of the bachelor thesis as well as an online version of that manual, located at http://www.winphone.cz.

Keywords

Silverlight, Windows Phone, XAML, C#, development, mobile applications, Microsoft, Mango, tutorial

Seznam použitých symbolů a zkratek

XAML – eXtensible Application Markup Language

WP – Windows Phone

WPF – Windows Presentation Foundation

XML – eXtensible Markup Language

XNA – platforma pro vývoj 2D a 3D

.**NET** – softwarová platforma společnosti Microsoft

VB – programovací jazyk Visual Basic

Obsah

1. Úvod 8
2. Pohled do historie platformy Windows Phone 9
3. Základní specifikace a filozofie operačního systému 11
   3.1. Metro design 11
   3.2. Hardwarové požadavky 12
   3.3. Architektura platformy Windows Phone 7 13
   3.4. Vývoj Windows Phone 7 aplikací 13
      3.4.1. Silverlight 13
      3.4.2. XNA 14
      3.4.3. Nástroje pro vývoj aplikací 14
   3.5. Životní cyklus projektu 15
   3.6. Životní cyklus aplikace 15
      3.6.1. Událost Launching 15
      3.6.2. Running 16
      3.6.3. Metoda OnNavigatedFrom 16
      3.6.4. Událost Deactivated 16
      3.6.5. Dormant 16
      3.6.6. Tombstoned 16
      3.6.7. Událost Activated 16
      3.6.8. Událost Closing 17
4. Představení nástrojů pro vývoj Windows Phone aplikací 18
   4.1. Visual Studio 18
   4.2. Integrovaný emulátor 22
   4.3. Expression Blend 22
5. Základy uživatelského rozhraní 24
   5.1. XAML 24
   5.2. Základní komponenty 24
      5.2.1. Textblock 24
      5.2.2. TextBox 25
      5.2.3. Image 25
      5.2.4. Slider 25
      5.2.5. Checkbox a RadioButton 25

5.2.6. HyperlinkButton 26
5.2.7. Button 26
5.2.8. ListBox 26
5.3. Pozicovací komponenty 27
5.3.1. Canvas 27
5.3.2. Grid 27
5.3.3. StackPanel 27
5.3.4. ScrollViewer 27
5.4. Zdroje a styly 28
5.5. Windows Phone komponenty 29
5.5.1. Panorama 29
5.5.2. Pivot 29
5.5.3. Orientace přístroje 30
5.5.4. Aplikační lišta 30
5.5.5. Dlaždice 31
6. Speciální funkce přístroje 34
6.1. Zvuk 34
6.2. Alarmy 34
6.3. Upomínky 34
6.4. Tasky 35
6.4.1. Launchers 35
6.4.2. Choosers 36
6.5. Izolované uložiště 36
7. Ukázková aplikace 38
7.1. Příklad 38
8. Doprovodný materiál 43
8.1. Příručka 43
8.2. Webová prezentace 43
9. Závěr 44
Literatura 45
Seznam obrázků 46
Přílohy 47

1. Úvod

Bakalářská práce má za cíl seznámit čtenáře se základními aspekty vývoje v technologii Silverlight pro mobilní operační systém Windows Phone 7 společnosti Microsoft. Veškeré ukázky kódu jsou napsány pomoci programovacího jazyka C#. Bakalářská práce je základní verzí rozsáhlejší příručky, která je součástí práce ve formě přílohy. Kromě toho je dostupná i online verze této příručky.

První kapitolou práce je letmý úvod do blízké historie operačního systému. Po historii následuje kapitola, která popisuje základní specifika a filozofii tohoto moderního operačního systému. V následující kapitole je popsáno základní programové vybavení, které je nezbytné pro efektivní vývoj pro tento operační systém. O základních dostupných komponentách pojednává pátá kapitola. V šesté kapitole jsou nastíněny dostupné speciální funkce přístroje, které můžeme při vývoji aplikaci využít. Sedmá kapitola demonstruje ukázkový příklad pro práci s webovými službami.

2. Pohled do historie platformy Windows Phone

Společnost Microsoft má více jak desetiletou tradici v oblasti mobilních systémů. V minulých letech a zvlášť pak v období od roku 2006 do roku 2007, byl jejich operační systém Windows Mobile velice oblíben, tomu odpovídal i výrazný tržní podíl. Operační systém Windows Mobile byl oblíben pro své bohaté možnosti všemožných úprav, nebyl vázáný na konkrétní hardware a dovoloval výrobcům zařízení si systém upravit podle svých představ. To se samozřejmě negativně odráželo na stabilitě systému a nejednotnosti platformy, takže testování aplikací byla pro vývojáře učiněná noční můra. Kromě toho všeho začal Microsoft ztrácet svůj podíl na trhu na úkor nové generace operačního systému iOS od Apple a operačního systému Android od Google, který začal nabývat velké oblibě. To vše vedlo Microsoft k radikálnímu kroku, ukončit vývoj Windows Mobile a vytvořit nový systém, který bude dostatečně konkurence schopný.

V roce 2010 na konferenci Mobile World Congress společnost oznámila nový mobilní operační systém Windows Phone 7 [8]. Windows Phone má znamenat novou éru v oblasti mobilních operačních systému společnosti Microsoft. Byly vydány minimální HW specifikace pro výrobce přístrojů, na kterých Windows Phone poběží. Tím chce Microsoft samozřejmě zajistit garanci plynulého běhu operačního systému na všech Windows Phone zařízeních. Také je zde výhoda pro tvůrce aplikací, protože se jim znatelně zjednoduší testování aplikací. Windows Phone také není zpětně kompatibilní s aplikacemi z operačního systému Windows Mobile.

Obrázek 1 Logo operačního systému Windows Phone

Microsoft také s novou verzí přinesl naprosto unikátní uživatelské rozhraní, které se nepodobalo ničemu, co bylo na trhu. Jednoduchost, správné použití typografie a zaměření na informace a intuitivní způsob, jak se k nim dostat, zaručovaly vysoce pohodlný a kvalitní uživatelský zážitek.

Od té doby vydal Microsoft několik aktualizací systému. První z aktualizací byla aktualizace pojmenovaná jako NoDo, která přinesla větší rychlost, podporu kopírování a vkládání textu, nové vlastnosti Marketplace a tak dále [1]. Další velkou aktualizací byla aktualizace

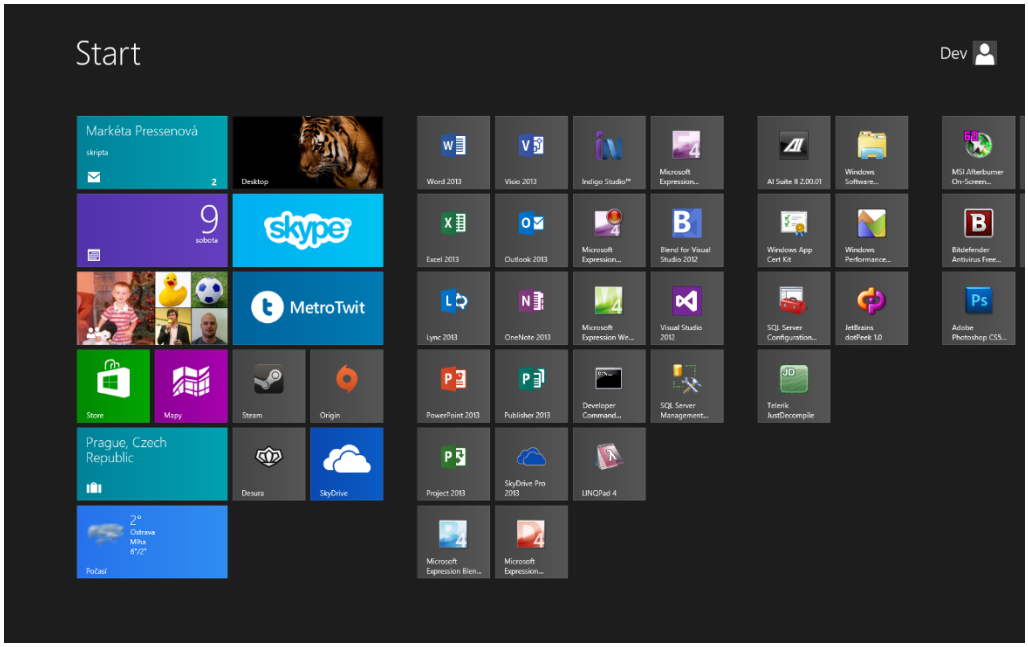
s označením Mango, která se do zařízení dostala v říjnu roku 2011. Windows Phone 7 se začínal označovat jako Windows Phone 7.5. Systém doznal znatelných aktualizací, jako je integrace internetového prohlížeče Internet Explorer 9, integrace sociálních služeb Twitter a LinkedIn, podpora češtiny a dále více jak 500 nových funkcí a vylepšení. Poslední znatelnou aktualizací, která pozvedla označení operačního systému na Windows Phone 7.8, přinesla hlavně přepracovanou startovací obrazovku systému a řadu jiných vylepšení. Tato aktualizace měla spíše jenom povznést starší generaci operačního systému na novou generaci, alespoň co se vizuální reprezentace týče [1].

V říjnu roku 2012 byla uvedena nová generace operačního systému označena pod číslovkou 8. Ta přináší vcelku radikální změny, jako je nahrazení architektury Windows CE s architekturou Windows NT či odstranění Silverlight a XNA frameworků. Kromě toho samozřejmě neexistuje zpětná kompatibilita mezi oběma generacemi a aplikace napsané pro Windows Phone 7 je nutné převést pro Windows Phone 8. Nicméně zkušenost, kterou získáte s vývojem pro starší generaci operačního systému, se výborně uplatní i pro vývoj pro novou generaci, jelikož filozofie pomocí jazyka XAML zůstává stejná.

3. Základní specifikace a filozofie operačního systému

3.1. Metro design

Microsoft zavádí do většiny svého produktového portfolia nový styl designu, který je, dá se říct, jedinečný na trhu. Nepodobá se ani jedinému stylu, který můžeme běžně vidět. Metro nebo chcete-li Microsoft design style, vsází na jednoduchost a eleganci. Hlavní důraz je kladen na samotný obsah (Content before Chrome) a na vhodné použití typografie, která může skvěle fungovat i jako designový prvek [2].



Obrázek 2 Ukázka Metro designu implementovaného v operačním systému Windows 8

Tento design se nejvíce dostal do podvědomí publika při uvedení právě zařízení s operačním systémem Windows Phone. Každopádně už před příchodem Windows Phone zařízení Microsoft tento styl designu začal implementovat do svých dalších produktů. V momentální době je tendence Microsoftu tento styl implementovat napříč produktovým portfoliem. Metro lze nalézt v nové verzi kancelářského balíku Office13, také již ve výše zmíněném operačním systému Windows 8 a také je tomuto designu přizpůsobován design webových stránek pro vývojáře.

3.2. Hardwarové požadavky

Každý výrobce zařízení, na kterém poběží mobilní operační systém Windows Phone 7, musí splnit minimální hardwarové specifikace deklarované Microsoftem.

Následující tabulku přehledně zobrazuje minimální hardwarové specifikace, které musí výrobce splnit.

Rozlišení WVGA (480x800) GPU DirectX9	Paměť 256 MB RAM Úložiště 4GB
CPU architektura ARM v7	Multitouch 4-bodový Hardwarové tlačítka Back, Start, Search
GPS Akcelerometr Kompas FM rádio	Digitální fotoaparát 5Mpx (nepovinný) WiFi (nepovinný)

Tabulka 1 Minimální hardwarové požadavky operačního systému Windows Phone [3]

Procesor by měl být postaven na architektuře ARM v7, přičemž jsou podporovány jenom čipy SnapDragon MSM7X30, MSM8X55, QSD8X50.

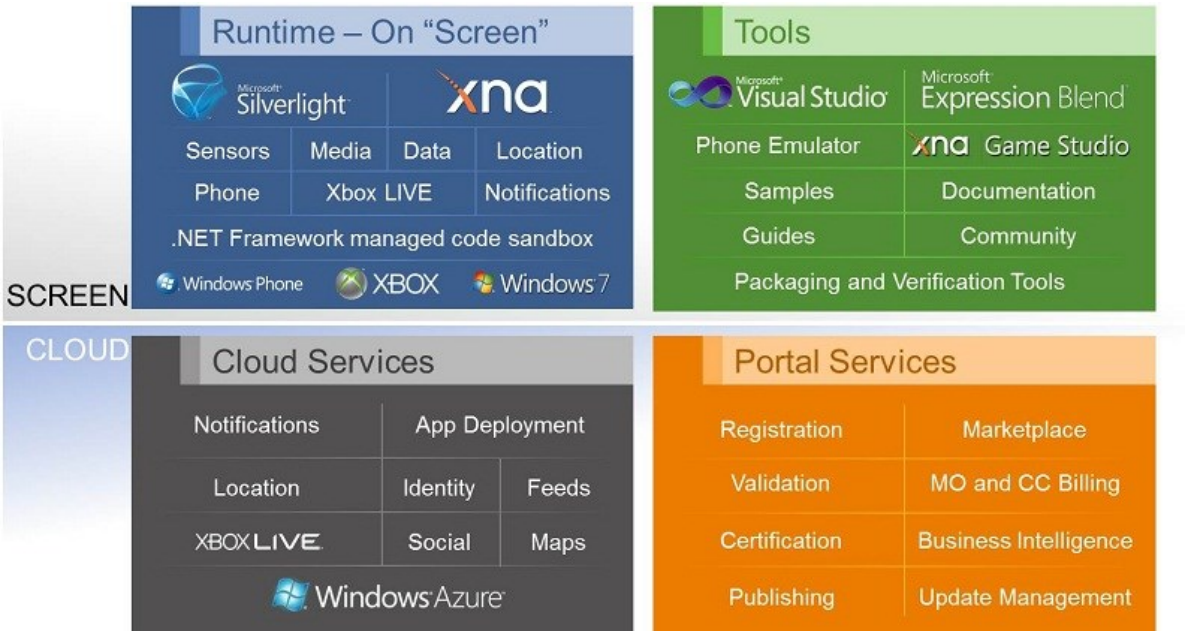
Paměť musí být alespoň o velikosti 256MB a interní úložiště o velikosti nejméně 4GB.

Každé WP7 zařízení bude mít na přední straně telefonu tři hardwarová tlačítka (Back, Start, Search), která budou popsána později.

Dalšími nezbytnostmi jsou pak podpora lokalizace telefonu (GPS), akcelerometr pro snímání pohybu, FM rádio.

Integrovaný digitální fotoaparát nemusí být povinným vybavením přístroje, jak tomu bylo u prvních verzí systému.

3.3. Architektura platformy Windows Phone 7



Obrázek 3 Architektura WP7 [5]

Architektura platformy Windows Phone 7 se skládá ze čtyř hlavních komponent.

- Runtime – On „Screen", kde jsou zahrnuty frameworky Silverlight a XNA + specifické vlastnosti pro WP7 jako je přístup k senzorům, lokalizaci apod.
- Tools – zde se řadí nástroje jako je Visual Studio, Expression Blend a k nim nainstalované potřebné nástroje. Tyto nástroje umožňují rychlý vývoj aplikací, jejich jednoduché testování a publikování.
- Cloud Services – umožňují sdílet data přes cloudové služby, řadí se tu služby, jako je Windows Azure, XBOX Live, lokalizační služby a jiné.
- Portal Services, zde se nachází Marketplace, který slouží pro distribuci aplikací.

3.4. Vývoj Windows Phone 7 aplikací

Microsoft pro vývoj aplikací staví na již existujících technologiích, takže je poměrně velká šance, že vývojář znající jednu z těchto technologií, se bude jednoduše adaptovat na vývoj pro WP7. Framework, který můžeme pro vývoj Windows Phone 7 aplikací použít, je XNA nebo Silverlight.

3.4.1. Silverlight

První z frameworků určených pro vývoj aplikací pro WP7 je Silverlight. Silverlight má své kořeny ve webovém vývoji, kde měl ve své době konkurovat technologiím, jako byla dobře známá a populární technologie Flash. Silverlight je totiž určen pro vývoj bohatých internetových aplikací, také známých pod zkratkou RIA (Rich Internet Application).

Silverlight využívá značkovacího jazyka XAML pro deklaraci uživatelského rozhraní. Silverlight je velice silný nástroj pro vytváření business aplikací s důrazem na bohaté uživatelské rozhraní. Pro aplikační logiku lze využít C#, VB popřípadě některý z dynamických jazyků (IronPython, IronRuby). Samozřejmostí je sada vytvořených komponent a možnost využití existujících. NET knihoven.

Programování aplikací pomocí technologie Silverlight je cílem této práce.

3.4.2. XNA

Pro vytváření 2D a 3D her nám nejlépe poslouží XNA framework. XNA framework není obsahem této práce, nicméně dobrým výchozím bodem je webová stránka http://create.msdn.com/en-us/education/roadmap. Bohužel v současné době už Microsoft XNA Framework nepodporuje. Nenalezneme jej tak v aktuální verzi operačního systému Windows Phone 8.

3.4.3. Nástroje pro vývoj aplikací

Všechny nástroje, které jsou nezbytné pro vývoj WP7 aplikací najdete na adrese https://dev.windowsphone.com/en-us. Na této webové adrese naleznete také množství tutoriálů a doporučení, které můžete využít při dalším studiu.

Nezbytným nástrojem pro vývoj WP7 aplikací je Visual Studio. Toto vývojové prostředí není nutné představovat, jelikož každý vývojář, programující aplikace pomocí Microsoft technologií, ho určitě důvěrně zná. Microsoft nabízí i bezplatnou edici tohoto vývojového prostředí. Tato edice je označena jako Express a nabízí se v různých variantách dle cílení vývoje. V Express edici se nalézá i Visual Studio Express pro Windows Phone vývoj (Visual Studio Express for Windows Phone). Aktuálně si můžeme vybrat ze dvou verzí. Visual Studio 2010 nebo Visual Studio 2012. Novější verzi použijeme v případě, že vlastníme operační systém Windows 8 v 64bitové edici. Visual Studio 2012 je také nezbytnou podmínkou pro to, pokud budeme v budoucnu cílit vývoj na platformu Windows Phone 8.

Výborným nástrojem pro tvorbu uživatelského rozhraní a komplexních animací je nástroj Expression Blend. Expression Blend pro návrh Windows Phone 7 aplikací je společně s dalšími nástroji dodáván v balíku Windows Phone Developer Tools. Tato verze Expression Blendu umožňuje vytvářet pouze aplikace určené pro WP7. Pokud jste ale studentem, můžete se ke kopii plné verze Expression Blendu, potažmo celého balíku Expression Studio, snadno dostat, a to přes programy jako je DreamSpark či MSDN AA program. Plná verze Expression Blend tak kromě návrhu uživatelského rozhraní pro WP7 podporuje i projekty založené na technologii Silverlight určené pro webové prostředí a WPF pro desktopové aplikace.

Windows Phone emulátor je integrován ve Visual Studiu a umožňuje nám simulovat skutečné zařízení. Samotný emulátor je velice schopný v simulování reálného zařízení, bohužel ne všechny funkce reálného zařízení jsou dostupné.

3.5. Životní cyklus projektu

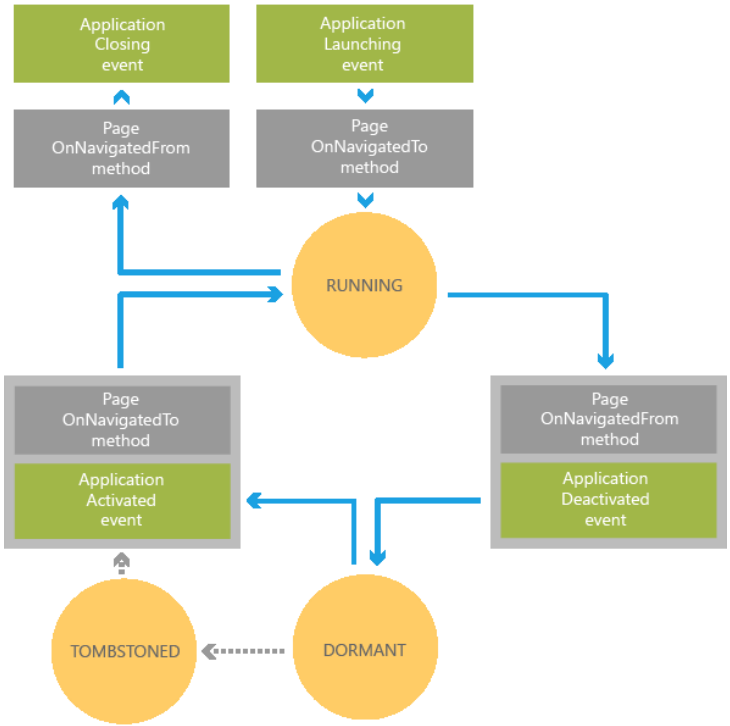
Prvním krokem bude zaregistrování na App Hubu pomocí Windows Live ID. Poté můžeme přes App Hub zaregistrovat až tři fyzická zařízení, která budou určena pro testování námi vytvořených aplikací.

Po vytvoření aplikace je nutné, aby naše aplikace prošla certifikačním řízením a mohla tak být umístěna na Marketplace. Dokument, který popisuje, základní kritéria pro získání certifikátu naleznete na této adrese http://go.microsoft.com/?linkid=9730558.

Pokud naše aplikace úspěšně prošla certifikačním řízením, pak je umístěna na Marketplace. Odtud už může náš potencionální zákazník aplikaci stáhnout a nainstalovat na své zařízení. Zákazník pak může umísťovat své hodnocení aplikace, komentovat ji apod. Je možné později nahrát aktualizovanou verzi, která bude mít opravené chyby, případně budou doplněny nové vlastnosti.

3.6. Životní cyklus aplikace

Aplikace se během svého životního cyklu nachází v různých stavech. Jednotlivé stavy jsou zobrazeny na obrázku.



Obrázek 4 Znázorněné stavy a události v životním cyklu aplikace

3.6.1. Událost Launching

Jakmile je aplikace spuštěna, vyvolá se událost Launching. Abychom zajistili rychlý start aplikace, měli bychom v rámci této události provést, co nejmenší množství logiky a

vyvarovat se časově náročným operacím jako práce se soubory nebo se síti. Tyto úkoly by měly být provedeny ve vlastním vlákně po naběhnutí aplikace.

3.6.2. Running

Po spuštění aplikace se stav změní na Running - běžící.

3.6.3. Metoda OnNavigatedFrom

Tato metoda je vyvolána na úrovni stránky, pokud se uživatel naviguje pryč z této stránky v rámci aplikace.

3.6.4. Událost Deactivated

Deactivated událost je vyvolána, když se uživatel naviguje pryč z aplikace stisknutím tlačítka Start nebo spuštěním jiné aplikace. Také je vyvolána, když aplikace spustí task typu Chooser.

V rámci obsluhy této události by měla aplikace uložit všechny neuložené aplikační data, takže mohou být obnoveny později, pokud bude potřeba.

Windows Phone aplikacím je k dispozici objekt State, což je datová struktura typu slovník (Dictionary), kterou lze použít pro uložení stavu aplikace.

3.6.5. Dormant

Když se uživatel naviguje pryč z aplikace a po vyvolání události Deactivated, operační systém se pokusí uvést aplikaci do stavu Dormant, tedy spící stav. V této fázi jsou všechny vlákna zastavena, nicméně aplikace zůstává zavedena v paměti.

Pokud jsou nové aplikace spuštěny po době, kdy aplikace byla uvedena do stavu Dormant a zároveň tyto aplikace vyžadují paměť pro hladký chod, operační systém začne uvádět spící aplikaci do stavu Tombstoned, aby získal nově uvolněnou paměť.

3.6.6. Tombstoned

Aplikace v tomto stavu byla ukončena, nicméně operační systém uchovává objekt State, který byl naplněn v průběhu Deactivated události. Zařízení bude udržovat takovou informaci pro maximálně pět aplikací běžící v témže okamžiku. Pokud byla aplikace ukončena a uživatel se naviguje zpět do aplikace, aplikace bude znovu spuštěna a může použít uložená data k obnovení svého stavu.

3.6.7. Událost Activated

Tato událost je volána, když se uživatel vrátí do aplikace ze stavu spící či ukončené aplikace. Aplikace by poté měla zkontrolovat vlastnost IsApplicationInstancePreserved ke zjištění, zda k obnovení došlo v rámci uspané aplikace či ukončené. Pokud je hodnota této vlastnosti True, pak došlo k navigaci ze spící aplikace a stav aplikace byl automaticky uchovaný operačním systémem. Pokud je False, aplikace byla ukončena a pro vyčtení stavu aplikace by se měl použít objekt State.

3.6.8. Událost Closing

Je vyvolána, když se uživatel naviguje zpět z první stránky aplikace. V tomto stavu je aplikace ukončena a žádný stav není uložen. V této události by měla aplikace uložit všechna svá data. Existuje ale limit 10 sekund pro každou událost. Pokud je tento limit překročen, pak je aplikace ukončena. Je tedy dobré ukládat data průběžně a vyhnout se těmto úkonům v rámci této události.

4. Představení nástrojů pro vývoj Windows Phone aplikací

Základním nástrojem, který budeme využívat při tvorbě WP7 aplikací, je vývojové prostředí Visual Studio. Existuje celá řada edicí tohoto prostředí, ale pro naše účely si vystačíme i s jeho bezplatnou edicí Visual Studio 2010 Express pro vývoj WP7 aplikací, případně Visual Studio 2012 Express pro Windows 8x64.

Pro stažení všech nezbytných nástrojů pro vývoj WP7 aplikací, navštivte App hub na adrese http://create.msdn.com/en-us/home/getting_started, kde se lze dostat na stažení všech potřebných nástrojů. Případně pro stažení určitých SDK (Software Development Kit) můžete rovnou navštívit stránku https://dev.windowsphone.com/en-us/downloadsdk. Na této stránce máte několik možností, které SDK si můžete stáhnout. SDK 7.1 představuje aktualizaci Windows Phone 7.5. SDK 7.1.1 představuje aktualizaci a verzi systému Windows Phone 7.5 Refresh. SDK Update for Windows Phone 7.8 je SDK s podporou pro poslední aktualizaci první generace operačního systému. Popis jednotlivých SDK a obsah najdete pod příslušným odkazem.

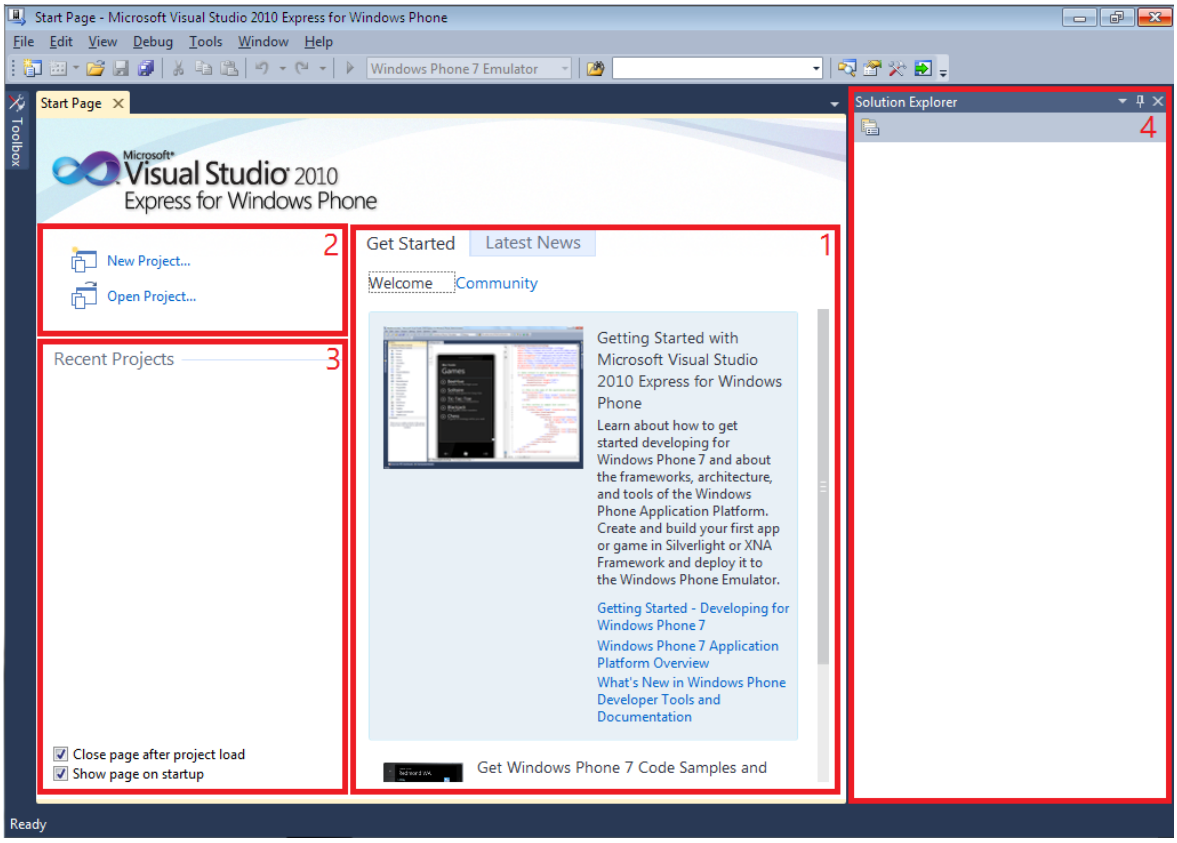
Nástroje a knihovny, které by měly být obsaženy v SDK 7.1, jsou uvedeny v následujícím seznamu:

- Microsoft Visual Studio 2010 Express for Windows Phone
- Windows Phone Emulator
- Windows Phone SDK 7.1 Assemblies
- Silverlight 4 SDK and DRT
- XNA Game Studio 4.0
- Microsoft Expression Blend SDK for Windows Phone 7

4.1. Visual Studio

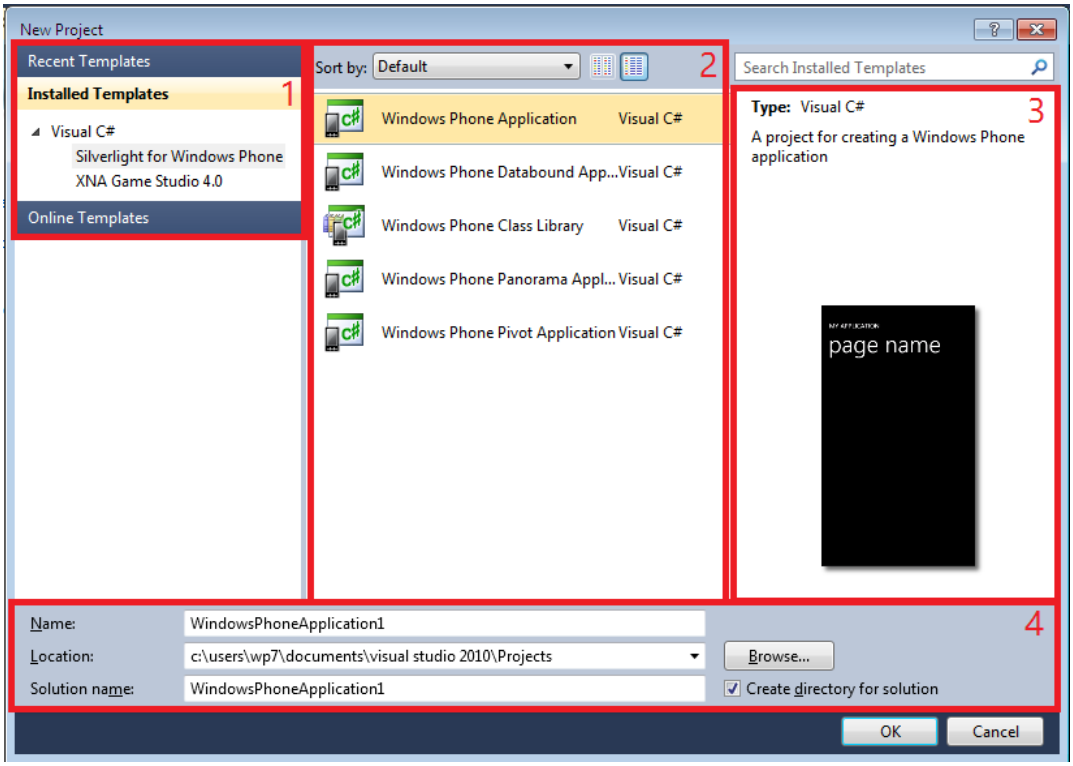
Spuštění Visual Studia provedeme přes jeho zástupce, Start > All programs > Microsoft Visual Studio 2010 Express > Microsoft Visual Studio 2010 Express for Windows Phone.

Pokud Visual Studio spouštíme a tedy neotvíráme existující projekt, zobrazí se nám tzv. Startovací obrazovka (Start Page). Startovací obrazovka obsahuje několik sekcí, které si blíže popíšeme.



Obrázek 5 Hlavní stránka vývojového prostředí Visual Studio 2010

- **1** Největší plochu zabírají informace určené pro Windows Phone vývojáře. Na záložce Get Started nalezneme užitečné informace o tom, jak začít programovat pro tuto platformu. Najdeme zde i doporučené průvodce pro správný návrh uživatelského rozhraní Metro, ukázkové kódy, a také odkazy na komunitní weby a blogy zajímající se o Windows Phone. Na záložce Latest News máme nakonfigurovanou RSS čtečku, která agreguje informace o posledních novinkách, které se na této platformě udály. Je vhodné sledovat tyto informace, protože velice často se zde objevují informace o nových aktualizacích a verzích nástrojů.
- **2** Zde naleznete rychlý přístup pro vytvoření nového projektu nebo otevření již existujícího projektu. Samozřejmě lze tyto úkony provést přes hlavní menu File > New Project… resp. File > Open Project…. Případně můžeme použít klávesových zkratek Ctrl+Shift+N resp. Ctrl+Shift+O.
- **3** Slouží pro rychlé otevření naposledy otevřených projektů. Jelikož jsme momentálně nevytvořili žádný projekt, tak tuto záložku máme prázdnou.
- **4** Solution Explorer je okno, které zobrazuje ve stromové struktuře všechny soubory a adresáře tvořící náš projekt.
- Pro vytvoření nové aplikace klikneme na File > New Project…, případně klikneme na New Project… ze startovací stránky. Po kliknutí se nám zobrazí následující dialog.

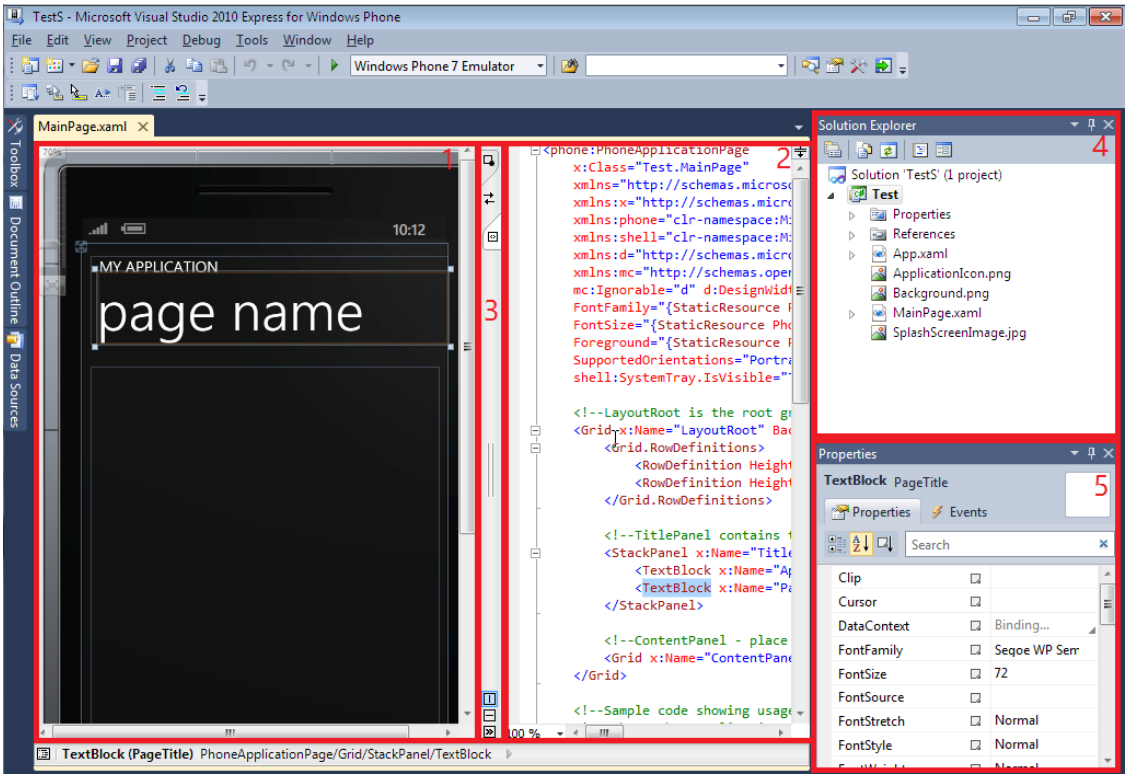


Obrázek 6 Výběr šablony pro vývoj Windows Phone aplikací ve vývojovém prostředí Visual Studio 2010

- **1** V této sekci je seznam všech nainstalovaných šablon. Visual C# obsahuje všechny šablony, které budou právě používat jazyk C# jako svůj primární programovací jazyk. Pro naše potřeby je nejdůležitější položka Silverlight for Windows Phone.
- **2** Zde máme možnost vybrat šablonu projektu dané technologie.
- **3** Informace o právě zvolené šabloně nalezneme právě na tomto místě, součástí popisu je také její náhled.
- **4** Je důležité správně pojmenovat svůj projekt a uložit ho na požadované místo. Tato část slouží právě pro tyto účely.

Pro testovací účely vybereme šablonu Windows Phone Application pro Silverlight, zvolíme si vhodné umístění a pojmenování aplikace a potvrdíme tlačítkem OK.

Po vygenerování všech potřebných souborů šablony jsme připraveni vyvíjet.. Abychom se ale v tomto prostředí neztratili, popíšeme si jednotlivé okna.



Obrázek 7 Popis vývojové prostředí Visual Studio 2010

- **1** a **2** Hlavní okno, kde vývojář tráví zpravidla nejvíce času, je rozděleno na dvě části.
- **1** Ve výchozím nastavení je v levé části okno, pro vizuální návrh uživatelského rozhraní aplikace. Můžeme tak vizuálně přidávat, přesouvat a různými způsoby modifikovat jednotlivé prvky. Všechny změny se přirozeně projeví i v XAML kódu.
- **2** V pravé části nalezneme okno s příslušným XAML kódem. XAML kód je značkovací jazyk určený pro návrh uživatelského rozhraní. Změny provedené v XAML kódu se projeví také v návrhové části.
- **3** Mezi těmito dvěma okny se nachází oddělovač s užitečnými vlastnostmi. Často při vývoji chceme pracovat pouze s jedním typem zobrazení. Pomoci oddělovačů můžeme skrýt jeden z panelů nebo měnit mezi horizontálním resp. vertikálním zobrazením panelů.
- Po kliknutí na symbol se nám aktivuje návrhové okno. Po dvojitém kliknutí je XAML editor skryt. Analogicky stejné chování platí pro symbol XAML editoru . Přehození oken lze realizovat přes tlačítko .
- Pro vertikální resp. horizontální rozdělení oken použijeme tlačítko resp. .
- Okno na pravé straně lze skrýt pomocí tlačítka a pro zpětné zobrazení stačí kliknout na symbol . Pro změnu poměru rozdělení oken lze realizovat tažením při zobrazeném symbolu . Tento symbol se zobrazí po přesunutí kurzoru nad tento oddělovač.
- **4** V okně Solution Explorer lze najít všechny soubory, ze kterých se naše aplikace skládá. Jednotlivé soubory budou popsány ve třetí kapitole.

- **5** Panel Vlastností (Properties) budeme používat velice často. Najdeme zde totiž vlastnosti jednotlivých komponent. Pokud máme vybranou určitou komponentu, pak v tomto panelu nalezneme její veškeré vlastnosti. Můžeme je tak jednoduše modifikovat. Pokud budeme chtít změnit text, který se nachází na komponentě TextBlock, tak jednoduše tuto komponentu vybereme a v panelu vlastnosti najdeme její hodnotu pro text, kterou následně zmodifikujeme. Změna se také projeví do XAML kódu. Kromě vlastností se tu nachází i položka pro Události (Events), kde můžeme přiřazovat jednotlivé obsluhující metody pro události dané komponenty. Například při kliknutí na tlačítko budeme chtít vyvolat událost Click a následně tuto událost vhodně obsloužit. Pokud se vám tento panel nezobrazil, tak jednoduše si ho zobrazíte přes nabídku View > Other Windows > Properties Window.
- V levé části lze najít panel Toolbox, který je zatím skryt. Tento panel obsahuje sadu standardních komponent jako je textové pole, tlačítko, komponenty rozvržení a jiné. Pro zobrazení dostupných komponent stačí nad panelem Toolbox umístit kurzor. Komponenty lze pak tažením přesunout do návrhu aplikace.

4.2. Integrovaný emulátor

Pokud vytváříme aplikaci pro WP7, tak je žádoucí, abychom si vyvíjenou aplikaci mohli nějakým způsobem otestovat a odladit. Aplikaci můžeme nahrát a otestovat buď na skutečné zařízení, nebo v rámci emulátoru, který je součástí dodávaného SDK a zároveň integrované ve Visual Studiu. Pokud aplikaci chceme nahrát na skutečné zařízení, tak musíme také splnit to, že budeme mít přístroj pro nahrávání aplikací odblokovaný pomoci vlastního vývojářského účtu.

Abychom si mohli emulátor otestovat, je nejprve nutné si ho spustit z Visual Studia. Pokud máme stále otevřenou aplikaci, pak stačí použit klávesu F5, což je standardní klávesa pro spuštění aplikace s možnosti ladění. Pokud v aplikaci nemáme žádné chyby, které by znemožňovaly spuštění aplikace, tak se nám spustí emulátor. Prvotní spuštění emulátoru zabere více času, následně ale emulátor nemusíme znova zavírat. Stačí, když aplikaci znovu spustíme pomoci klávesy F5 nebo Ctrl + F5 (bez možnosti ladění) a tím se vyhneme dlouhému nahrávání emulátoru při spuštění nové instance.

4.3. Expression Blend

Microsoft Expression Blend se řadí do rodiny produktů souhrnně označených jako Expression Studio. Balík nástrojů Expression Studio je určen především pro designéry a webové vývojáře.

Expression Studio obsahuje řadu nástrojů, které pomáhají při tvoření bohatého uživatelského obsahu aplikace. Expression Design je pro tvorbu grafiky, kterou můžeme dále použít v rámci naši aplikace. Expression Web je nástroj určený pro tvůrce webu, obsahuje navíc rozšíření SuperPreview, která generuje náhledy v různých prohlížečích. Pro nás je ovšem nejdůležitější nástroj Expression Blend, který nám v mnoha případech ušetří velké množství času při návrhu uživatelského rozhraní aplikace.

Expression Blend je určen pro rychlý a pohodlný návrh uživatelského rozhraní pro Silverlight a WPF aplikace. Možná si lámete hlavu, proč použít nástroj Expression Blend, když to samé dokáže zvládnout i integrované návrhové prostředí ve Visual Studiu? Síla Expression Blendu ale tkví hlavně v jednoduchém tvoření animací a definování komplexních stylů.

Expression Blend určený pro vývoj Windows Phone aplikací je nainstalován v rámci SDK. Tato verze Blendu je omezena pouze na vývoj Windows Phone aplikací. Pokud bychom chtěli pracovat s jiným projekty, jako je Silverlight určený pro web nebo aplikace napsané pomocí technologie WPF, museli bychom si vyšší verzi zaplatit.

5. Základy uživatelského rozhraní

5.1. XAML

Pro tvorbu uživatelského rozhraní se v Silverlightu používá značkovací jazyk XAML. Tento jazyk je používán i u jiných technologií jako je Windows Presentation Foundation pro desktopové aplikace nebo pro nové Windows 8 Metro aplikace.

XAML neboli eXtensible Application Markup Language, je tedy deklarativní jazyk sloužící pro popis uživatelského rozhraní. Jazyk XAML vychází z XML a je tedy dodržovaná hierarchická struktura, kdy existuje jen jeden kořenový element, zavádí se jmenné prostory (namespaces), definují se atributy na elementech nebo že rozlišuje malá a velká písmena (case-sensitive) [7].

Jednotlivé elementy, které jsou definovány v jazyce XAML, jsou převedeny na odpovídající instance objektů v. NET Frameworku. Hierarchie, které je definována pomocí jazyka XAML je do paměti převedena ve stejném formátu.

V kořenovém elementu se nachází řada jmenných prostorů. Základní jmenný prostor xmlns deklaruje to, že se jedná o XAML dokument. Jmenný prostor s prefixem x deklaruje další elementy resp. vlastnosti, které jsou specifické pro XAML. Prefix x může být změněn na cokoliv jiného, ale není to doporučeno, jelikož prefix x je standardně používaný a uznávaný.

Kromě výše uvedených jmenných prostorů můžeme jazyk XAML rozšířit i o jiné .NET typy.

Obecně platí, že každý XAML soubor má definovanou svou parciální třídu, která se označuje jako kód na pozadí (Code Behind). Nejčastěji je to soubor, který má navíc příponu *.cs nebo *.vb a to dle zvoleného programovacího jazyka. Přípona *.cs je pro C# a *.vb pro Visual Basic. V tomto kódu lze psát obsluhující logiku pro XAML, například obsluhu události kliknutí na tlačítko. Nicméně tato technika není v praxi doporučována a místo ní se zavádí návrhový vzor MVVM, kde logika je definována v samostatném souboru.

5.2. Základní komponenty

Technologie Silverlight sebou přináší řadu již vytvořených komponent pro uživatelské rozhraní. Vývojář tak nemusí trávit spoustu času tím, že by základní komponenty jako tlačítko či textové pole musel vytvářet. A díky bohatým možnostem úprav jednotlivých komponent a jejich stylů, si může každý vývojář komponentu upravit podle svých potřeb. Kromě základních komponent, které nabízí Silverlight, existuje i speciální sada komponent výhradně určena pro Windows Phone. Kromě těchto dodávaných komponent se vývojáři mohou setkat s řadou různých komponent, které vytvořila komunita.

5.2.1. Textblock

TextBlock je základní komponenta pro reprezentaci textové hodnoty. Její použití je velice jednoduché a prakticky vyžaduje nastavení jenom atributu Text. Samozřejmě je možné nastavit celou řadu jiných atributů, které se vážou na element TextBlock, jako je velikost textu, barva písma apod..

```
<TextBlock    Text="¡Visca Barça!"
              FontSize="48"
              TextAlignment="Center"
              Foreground="Blue">
</TextBlock>
```

5.2.2. TextBox

Většina aplikací, které přijímají vstup od uživatele, se neobejdou bez příslušných komponent. Samozřejmě i Silverlight nabízí komponenty, které nám s tímto běžným a častým úkolem pomohou. Uživatel, který zadává hodnoty do vstupních polí, tak nejčastěji dělá přes integrovanou softwarovou klávesnici, kterou operační systém nabízí. Existují i přístroje, které mají zvlášť zabudovanou hardwarovou klávesnici, těch je ale jenom mizivá většina. Softwarová klávesnice (Software Input Panel - SIP) může nabývat řady podob, přesně podle toho, jaký vstup od uživatele očekáváme.

Vstup a tedy typ klávesnice, který očekáváme, můžeme nastavit pomocí atributu InputScope.

```
<TextBox InputScope="Text" />
```

5.2.3. Image

Jak už název napovídá, tak komponenta Image slouží k výhradně k zobrazení obrázků. Omezení na datový typ obrázku je, aby byl ve formátu PNG nebo JPG. Ostatní formáty nejsou podporovány. Komponenta Image obsahuje atribut Source, kterým se definuje cesta k obrázku, který chceme zobrazit. Zdrojem může být obrázek, který se nachází někde na serveru a je definován adresou.

```
<Image Source="http://bit.ly/YceKek" />
```

5.2.4. Slider

Je komponenta, která vizuálně indikuje hodnotu v rámci stupnice. Obsahuje atributy pro nastavení maximální hodnoty stupnice – Maximum a pro nastavení minimální hodnoty stupnice – Minimum. Atribut Value udává aktuální hodnotu na stupnici.

```
<Slider Maximum="100" Minimum="1" Value="50" />
```

5.2.5. Checkbox a RadioButton

Chceme-li uživateli umožnit výběr jedné z možností, použijeme pro tento účel komponentu RadioButton. RadioButton obsahuje dva důležité atributy. GroupName slouží pro identifikaci, do které skupiny daný RadioButton patří a ze které skupiny se bude vybírat. Atribut Content nastavuje pouze obsah RadioButtonu.

```
<StackPanel>
        <RadioButton GroupName="G1" Content="Opt1" />
        <RadioButton GroupName="G1" Content="Opt2" />
        <RadioButton GroupName="G1" Content="Opt3" />
</StackPanel>
```

Pokud chceme uživateli poskytnout na výběr více než jednu možnost, pak použijeme komponentu CheckBox.

```
<StackPanel>
        <CheckBox Content="Opt1" />
        <CheckBox Content="Opt2" />
        <CheckBox Content="Opt3" />
</StackPanel>
```

5.2.6. HyperlinkButton

Pro navigaci mezi ostatními XAML stránkami slouží komponenta HyperlinkButton, kde cíl navigace se specifikuje pomocí vlastnosti NavigateUri. Komponenta se nedá použít pro navigaci na HTML stránku, nýbrž jenom na navigaci na jinou XAML stránku.

```
<HyperlinkButton NavigateUri="/Page.xaml" Content="Naviguj me" />
```

5.2.7. Button

Komponenta Button je základní komponentou, pomocí které uživatel instruuje naši aplikaci, aby něco vykonala. Pokud se tlačítko stiskne, pak je vyvolána příslušná událost, která je navázána na vlastnost Click. Komponenta Button se řadí do komponent, které obsahují vlastnost Content. Tato vlastnost může v sobě obsahovat libovolný XAML kód. To sebou přináší řadu výhod při definici uživatelského rozhraní. Tlačítko tak kromě prostého textu může obsahovat například obrázek.

```
<Button>
        <Button.Content>
                <StackPanel>
                        <TextBlock Text="Popisek Tlacitka" />
                        <Image Source="images/tiger.jpg" />
                </StackPanel>
        </Button.Content>
</Button>
```

5.2.8. ListBox

Pro zobrazení seznamu objektů můžeme použít komponentu ListBox. Komponenta ListBox obsahuje atribut ItemsSource, kde se vážou objekty, které se v seznamu mají zobrazit. Hodnotou může být jakákoliv kolekce, která implementuje rozhraní IEnumerable nebo IList.

```
<ListBox x:Name="SeznamLidi"></ListBox>
```

Důležitý je kód na pozadí, kde do ListBoxu, který je pojmenován SeznamLidi, předáváme pole jmen lidí.

```
// Constructor
public MainPage()
{
        InitializeComponent();

        // Napln seznam polem stringu
        SeznamLidi.ItemsSource = new [] {"Martin", "Marketa", "David",
        "Larry",Vlada", "Helena", "Zaneta", "Radka", "Marcel", "Jiri"};
}
```

5.3. Pozicovací komponenty

Pozicovací komponenty tvoří základní stavební kámen našeho uživatelského rozhraní. Pomocí těchto pozicovacích komponent rozmísťujeme vnořené komponenty určitým způsobem.

5.3.1. Canvas

Canvas nám nabízí největší svobodu, co se týká pozicování komponent. Komponenty se totiž pozicují podle daných souřadnic. Těmito souřadnicemi jsou atributy Canvas.Left a Canvas.Top. Dále obsahuje ZIndex, který určuje překrytí dvou komponent. Komponenty s větším číslem překrývají ty s menším číslem.

5.3.2. Grid

Grid využívá tabulkový layout. Pomocí vlatností ColumnDefinitions resp. RowDefinitions definuje jednotlivé sloupce resp. řádky. Umožňuje také sloučení sloupců resp. řádků pomocí vlastností ColumnSpan resp. RowSpan.

Při nastavování velikosti sloupce resp. řádku se můžeme setkat se třemi druhy zápisu těchto hodnot. Velikost můžeme nastavit na Auto, přičemž se velikost přizpůsobí obsahu. Můžeme být přesní a nastavit velikost přesně v pixelech. To se zadává číselnou hodnotou. A v neposlední řadě tu je znak *, který proporcionálně vyplní obsah komponenty Grid.

5.3.3. StackPanel

StackPanel nám umožňuje řadit prvky buď vertikálně, nebo horizontálně. StackPanel má výchozí řazení vertikálně. Prvky jsou řazeny v přesném pořadí, jak jsou definovány.

5.3.4. ScrollViewer

Umožňuje zobrazit obsah, který svou velikostí přesahuje definovaný kontejner. ScrollViewer tento obsah zobrazuje pomocí posuvníků.

5.4. Zdroje a styly

Pokud naše aplikace využívá stejné hodnoty různých atributů na různých místech aplikace, tak je vhodné vytvořit pro něj tak zvané Resources neboli česky zdroje. Těmito hodnotami můžeme nejčastěji myslet například barvy, které používáme v rámci aplikace, můžou to být různé typy fontů apod.. Resources jsou jednoznačně identifikované pomocí atributu Key s prefixem x.

Každý prvek, který dědí ze třídy FrameworkElement podporuje kolekci Resources, které jsou typu ResourceDictionary. Pokud například budeme definovat kolekci Resources v rámci elementu PhoneApplicationPage, pak všechny elementy, které se nacházejí uvnitř tohoto elementu, můžou dané Resources využívat.

```
<phone:PhoneApplicationPage.Resources>
               <SolidColorBrush x:Key="MyBrush"
                                Color="Red" />
</phone:PhoneApplicationPage.Resources>

<Grid x:Name="LayoutRoot"  Background="Transparent">
       <TextBlock Text="Barva za pouziti Resource"
                  Foreground="{StaticResource MyBrush}" />
</Grid>
```

Odkazovat se na Resources můžeme pomocí klíčového StaticResource a název Resource. StaticResource prohledá celý XAML dokument a pokud nenarazí na danou Resource, pokračuje v hledání dále mimo dokument XAML. Tím místem, kde pokračuje v hledání, je App.xaml soubor, kde taktéž mohou být definovány Resources, které mohou být aplikovány napříč aplikací.

Styl je v podstatě kolekce předdefinovaných vlastností, které se aplikují na daný prvek. Pomocí stylu tak můžeme dosáhnout konzistentního vzhledu napříč aplikací. Pokud se v budoucnu rozhodneme pro změnu konkrétní vlastnosti, tak bude nám stačit tuto změnu provést na jednom místě.

```
<Style TargetType="TextBlock"
       x:Key="TextBlockStyle">
               <Setter Property="FontSize" Value="48" />
               <Setter Property="FontWeight" Value="Bold" />
</Style>
```

Styl vyžaduje definici, na který prvek se má aplikovat pomocí vlastnosti TargetType a poté obsahuje množinu vlastností definované elementem s názvem Setter, kde se nastavují jednotlivé vlastnosti daného prvku. Abychom mohli aplikovat styl na prvek, musíme nastavit přes vlastnost prvku Style daný styl.

5.5. Windows Phone komponenty

5.5.1. Panorama

Komponenta Panorama se řadí mezi komponenty určené speciálně pro zařízení s operačním systémem Windows Phone.

Panorama panel vytváří tzv. panorama režim, kdy se zobrazuje několik sekcí vedle sebe a uživatel se na ně může podle libosti přesouvat. Díky tomu se značně eliminujeme i omezená velikost displeje přístroje.

Pokud vytváříme nový projekt pro Windows Phone, v nabídce šablon najdeme i připravenou šablonu pro aplikaci, která využívá Panorama komponenty.

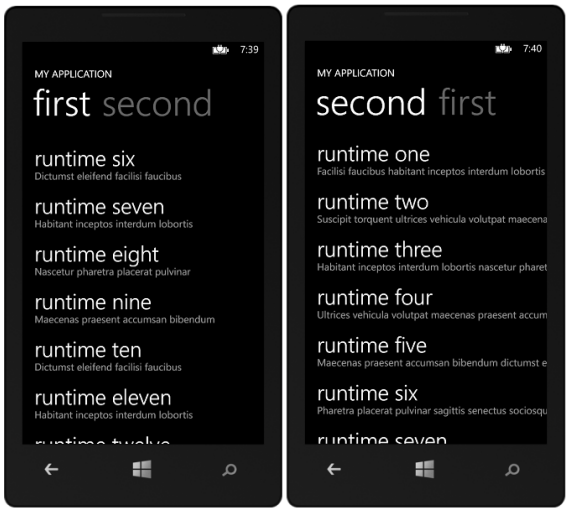
Obrázek 8 Defaultní vzhled komponenty Panorama

5.5.2. Pivot

Podobně jako komponenta Panorama i Pivot je speciální komponentou určenou pro zařízení Windows Phone.

Podobně jako Panorama nabízí jednotlivé sekce, na které se uživatel může přesouvat. Nicméně Pivot je přizpůsoben k zobrazení velkého množství dat. Na rozdíl od Panoramy, kde je určité překrytí, Pivot zaplňuje celou šířku stránky. Při práci s Pivot komponentou bychom se měli zamyslet na načítání dat, nastane-li potřeba (On Demand) a vyhnout se nahrání všech dat při startu aplikace.

Pokud vytváříme nový Windows Phone projekt v rámci Visual Studia, tak máme na možnost výběru Windows Phone Pivot App šablonu. Tento typ projektu vytvoří základní strukturu budoucí Pivot aplikace.



Obrázek 9 Defaultní vzhled komponenty Pivot

5.5.3. Orientace přístroje

Mobilní zařízení, na kterém běží operační systém Windows Phone, nabízí tři druhy orientace přístroje. Orientace přístroje na výšku – Portrait nebo na šířku – Landscape (Left, Right).

Abychom nastavili, které orientace naše aplikace bude podporovat, musíme vhodně nastavit atribut SupportedOrientations elementu PhoneApplicationPage. Hodnoty může nabývat Landscape, Portrait nebo PortraitOrLandscape. Pomoci poslední hodnoty nám bude aplikace podporovat obě orientace.

K identifikaci změny orientace třída PhoneApplicationPage obsahuje událost OrientationChanged, která je vyvolána v případě, že dojde ke změně orientace přístroje.

5.5.4. Aplikační lišta

Aplikační lišta funguje jako nástrojová lišta a také jako základní menu. Na aplikační liště můžeme mít grafické vyjádření jednotlivých akcí ve formě ikony nebo můžeme mít prosté textové popisky. Ikony na aplikační liště jsou permanentně na ploše zobrazeny. Pro jejich popis a další textové menu prvky stačí kliknout na symbol tří teček.

Aplikační lišta tedy může obsahovat ikony reprezentované objektem ApplicationBarIconButton a textové hodnoty ve formě ApplicationBarMenuItem. Samozřejmě mohou být mezi sebou libovolně kombinované.

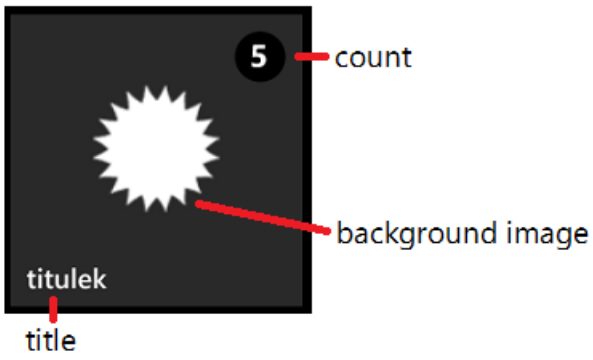
```
<phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar  IsMenuEnabled="True"
                               IsVisible="True">
                <shell:ApplicationBarIconButton       Text="search"

        IconUri="images/search.png" />
                <shell:ApplicationBar.MenuItems>
                        <shell:ApplicationBarMenuItem Text="save as" />
                        <shell:ApplicationBarMenuItem Text="export as" />
                </shell:ApplicationBar.MenuItems>
        </shell:ApplicationBar>
</phone:PhoneApplicationPage.ApplicationBar>
```

5.5.5. Dlaždice

Dlaždice fungují jako odkazy na příslušné aplikace. Svůj obsah mohou měnit a velmi efektivní cestou tak informovat uživatele o změnách, která nastaly v aplikaci resp. na serveru. Například u poštovního klienta můžu očekávat počet nepřečtených zpráv. Tyto dlaždice musí být ale nastaveny na hlavní obrazovce systému. Na hlavní obrazovku mohou být dlaždice připnuty tak, že uživatel dlouze podrží ikonu aplikace v seznamu nainstalovaných aplikací a vybere odkaz „pin to start“.

Živá dlaždice se skládá ze třech vrstev. Title, který by měl být názvem aplikace, Count, který zobrazuje informaci o stavu a BackgroundImage, který by měl být dostatečně jednoduchý, ale zároveň reprezentativní [9].



Obrázek 10 Živá dlaždice a její struktura

Jednotlivé vrstvy dané dlaždice jsou definovány v souboru WMManifest.xml.

```
<TemplateType5>
          <BackgroundImageURI IsRelative="true" IsResource="false">
                                        Background.png</BackgroundImageURI>
          <Count>5</Count>
          <Title>titulek</Title>
</TemplateType5>
```

Aplikace má pouze jednu hlavní živou dlaždici. Abychom s dlaždici mohli pracovat v rámci kódu, musíme použít vlastnost ActiveTitle třídy ShellTile. Tato vlastnost vrací výčet všech dlaždic v aplikaci. Pomocí metody First() získáme výchozí dlaždici.

```
var defaultTile = ShellTile.ActiveTiles.First();
```

Tuto výchozí dlaždici můžeme v rámci kódu aktualizovat. Abychom ji mohli aktualizovat, musíme si vytvořit novou instanci třídy StandardTileData, která bude obsahovat hodnoty, které budou aktualizovat dlaždici.

```
var defaultTile = ShellTile.ActiveTiles.First();

var updatedTile = new StandardTileData()
{
        Title = "novy titulek",
        Count = 99
};

defaultTile.Update(updatedTile);
```

Vytvořili jsme si novou instanci StandardTileData, kde jsme nastavili aktualizované informace. Poté na dlaždici, kde chceme data aktualizovat, zavoláme metodu Update, která jako parametr přijímá ShellTileData, přičemž StandardTileData jsou typu ShellTileData.

Pokud dlaždice nebyla připnuta na hlavní obrazovku, tak po připnutí se použije dlaždice s těmito novými informacemi namísto těch, které se nacházejí v souboru WMAppManifest.xml

Obrázek 11 Ukázka dlaždice

Aplikace může mít pouze jednu hlavní dlaždici. Může však mít různý počet tzv. sekundárních dlaždic (Secondary Tiles). Sekundární dlaždice se zpravidla vytvářejí z běžící aplikace. Po vytvoření dojde k okamžitému přepnutí na hlavní obrazovku operačního systému, kde dojde k zobrazení nově vzniklé dlaždice. Nemůžeme tedy najednou vytvořit několik dlaždic najednou, ale lze vytvořit postupně několik dlaždic navigující do téže aplikace. Sekundární dlaždice jsou tedy zástupci na určitou obrazovku naši aplikace.

Vytvoření nové dlaždice uděláme opět podle metody ShellTile.Create(), která jako parametr přijímá Uri, kde se bude zástupce odkazovat, a ShellTileData, což je samotný objekt dlaždice pro vytvoření.

```
var newTile = new StandardTileData
{
        Title = "DeepLink",
        Count = 11,
};

ShellTile.Create(new Uri("/DeepPage.xaml", UriKind.Relative), newTile);
```

Vytvoření dlaždice jsme přidali do obsluhy události po kliknutí na tlačítko. Jakmile uživatel stiskne tlačítko, dojde k tomu, že je navigován na hlavní stránku operačního systému a je mu zobrazena nově vzniklá dlaždice.

6. Speciální funkce přístroje

6.1. Zvuk

Chceme-li v přístroji přehrát nějaký zvukový soubor, můžeme k tomuto účelu použít třídu MediaElement. MediaElement se definuje v rámci XAML kódu, nemá však žádné uživatelské rozhraní, a proto není v rámci uživatelského rozhraní vidět.

```
<MediaElement x:Name="Sound" AutoPlay="False" Source="/zvuk.wma" />
```

6.2. Alarmy

Alarmy (Alarms) umožňují notifikovat uživatele ve formě zprávy o nastalých událostech v aplikaci. Uživatel může na tuto zprávu kliknout a po kliknutí bude navigován do aplikace. Alarm může uživatel také uspat (snooze) nebo „odmítnout“ (dismiss). Daný alarm může definovat i zvukovou stopu, která má být přehrána v případě spuštění.

```
Alarm alarm = new Alarm(Guid.NewGuid().ToString())
            {
                BeginTime = DateTime.Parse("2013-5-2T20:56"),
                Content = "Obsah alarmu"
            };

ScheduledActionService.Add(alarm);
```

Nezbytným krokem je zaregistrováni nově vzniklého alarmu do služby ScheduledActionService.

6.3. Upomínky

Upomínky (Reminders) jsou ve své podstatě velice podobné alarmům. Navíc oproti alarmům lze definovat i vlastní titulek. Z běžné funkcionality může uživatel zadat na jak dlouho dobu se má upomínka uspat (snooze) a po stisknutí na upomínku se mohou taktéž posílat data do aplikace, která je vlastníkem upomínky. Co není v rámci upomínky možné, je definice vlastního zvukového souboru.

Při vytváření upomínky postupujeme velice podobně jako v případě alarmu.

```
Reminder reminder = new Reminder(Guid.NewGuid().ToString())
            {
                BeginTime = DateTime.Now.AddMinutes(1),
                Title = "Upominka",
                Content = "Obsah upominky",
                NavigationUri = new Uri("/MainPage.xaml", UriKind.Relative)
            };

ScheduledActionService.Add(reminder);
```

6.4. Tasky

Operační systém Windows Phone obsahuje velké množství funkcionality řešící různé specifické úkoly. Například se může jednat o práci s Bing mapami, přístup ke kameře apod.. Aby mohla aplikace využívat těchto funkcionalit, nabízí Windows Phone tzv. tasky, chcete-li úlohy. Tyto tasky dále dělíme na typ Launcher a Chooser. Prvně jmenována skupina je založena na tom, že je uživatel navigován do jiné části systému s cílem splnění nějaké úlohy. Například zobrazení webového prohlížeče. Naopak Choosers slouží pro získání určitých informací. Potřebujeme vybrat emailovou adresu, pak použijeme příslušný Chooser.

6.4.1. Launchers

BingMapTask	Zobrazí se Bing mapy na základě vyhledávání či přímého zadání souřadnic.
EmailComposeTask	Zobrazí se okno pro vytvoření nové emailové zprávy.
MarketplaceSearchTask	Zobrazí se vyhledávací stránka pro Marketplace.
PhoneCallTask	Zobrazí stránku pro uskutečnění nového hovoru.
SearchTask	Zobrazí stránku pro hledání informací v rámci přístroje.
WebBrowserTask	Zobrazí integrovaný webový prohlížeč.
SmsComposeTask	Zobrazí stránku pro vytvoření nové SMS zprávy.

Tabulka 2 Přehled základních Launchers [10]

Vytvoření nového tasku typu Launcher je velice podobné pro všechny typy. Vytvoří se instance daného Launcher, nastaví se jeho vlastnosti a následně se zavolá metoda Show().

```
var emailTask = new EmailComposeTask();
emailTask.To = "recepient@email.com";
emailTask.Subject = "Secret message";
emailTask.Body = "This is a very secret content";

emailTask.Show();
```

6.4.2. Choosers

AddressChooserTask	Umožňuje výběr adresy uživatele ze seznamu.
CameraCaptureTask	Spustí kameru pro získání obrázku z kamery a daný obrázek je navrácen zpět do aplikace.
EmailAddressChooserTask	Umožňuje výběr existující emailové adresy a tato adresa je navrácena do aplikace.
PhoneNumberChooserTask	Umožňuje výběr telefonního čísla ze seznamu a navrátí jej do aplikace.
SavesContactTask	Umožňuje uložit nový kontakt do adresáře.
SaveEmailAddressTask	Umožňuje uložit emailovou adresu do adresáře a nově uložený email je navrácen do aplikace.
SavePhoneNumberTask	Umožňuje uložit nové telefonní číslo do adresáře a nově uložené číslo je navráceno zpět do aplikace.

Tabulka 3 Přehled základních Choosers[10]

Choosers a práce s nimi probíhá velice podobně jako s Launchers. Jediný rozdíl je v tom, že po vykonání se vrací nově vzniklé data. Každý Chooser obsahuje událost Completed, která je vyvolána, jakmile Chooser je ukončen.

V události je nutné zkontrolovat, zda Chooser byl úspěšně dokončen či nikoliv. To lze zkontrolovat pomoci vlastností Error a TaskResult, přičemž Null hodnota u vlastnosti Error a výčtová hodnota TaskResult.Ok indikují to, že Chooser proběhl úspěšně.

```
var phoneChooser = new PhoneNumberChooserTask();

phoneChooser.Show();

phoneChooser.Completed += (s, e) => {
      string phoneNumber = e.PhoneNumber;
};
```

6.5. Izolované uložiště

Každá aplikace má přidělenou privátní část v souborovém systému operačního systému. Všechny data, která jsou uložena v rámci tohoto privátního úložiště, jsou výhradně určena pro danou aplikaci a žádná jiná aplikace nemá k těmto datům přístup. Jakmile si uživatel nainstaluje aplikaci, je automaticky pro ni alokováno místo v souborovém systému. Naopak,

pokud uživatel aplikaci odinstaluje, tak i všechny data, která se nachází v tomto izolovaném úložišti, jsou smazána.

Pro práci s izolovaným uložištěm (Isolated Storage) slouží třída IsolatedStorageFile ze jmenného prostoru System.IO.IsolatedStorage. Třída IsolatedStorageFile obsahuje statickou metodu GetUserStoreForApplication(), pomoci které získáme přístup do izolovaného úložiště aplikace. Třída IsolatedStorageFile implementuje rozhraní IDisposable a proto je vhodné ji používat společně s klauzulí using.

Pomocí IsolatedStorageFile třídy můžeme vytvářet resp. číst soubory nebo adresáře z izolovaného uložiště. Kód pro vytvoření nového souboru v izolovaném úložišti by mohl vypadat následovně.

```
using (IsolatedStorageFile isf =
                IsolatedStorageFile.GetUserStoreForApplication())
{

}
```

7. Ukázková aplikace

7.1. Příklad

V následujícím příkladu se pomoci webových služeb připojíme k službě, která umožňuje konvertovat teplotu ve stupních Celsia na odpovídající stupně Fahrenheita a naopak. Tato webová služba se nachází na adrese http://www.w3schools.com/webservices/tempconvert.asmx.

- Vytvoříme si nový projekt, kde vybereme šablonu *Windows Phone App* pod jazykem C#. Vhodně projekt pojmenujeme a potvrdíme tlačítkem OK.
- Po vygenerování projektu otevřeme hlavní okno aplikace *MainPage.xaml*.
- Provedeme menší úpravy s textem, který se zobrazí na hlavní stránce aplikace, aby odpovídal našemu řešení problému.

```
<!--TitlePanel contains the name of the application and page title-->
<StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
          <TextBlock x:Name="ApplicationTitle" Text="Konvertor"
                     Style="{StaticResource PhoneTextNormalStyle}"/>
          <TextBlock x:Name="PageTitle" Text="Teplota"
                     Margin="9,-7,0,0"
                     Style="{StaticResource PhoneTextTitle1Style}"/>
</StackPanel>
```

- Do hlavní obrazovky resp. do komponenty Grid přidáme komponentu TextBox, která bude sloužit jako vstup pro zadání stupně teploty.

```
<!--ContentPanel - place additional content here-->
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
        <StackPanel>
                <TextBox x:Name="ZadanaTeplota" />
        </StackPanel>
</Grid>
```

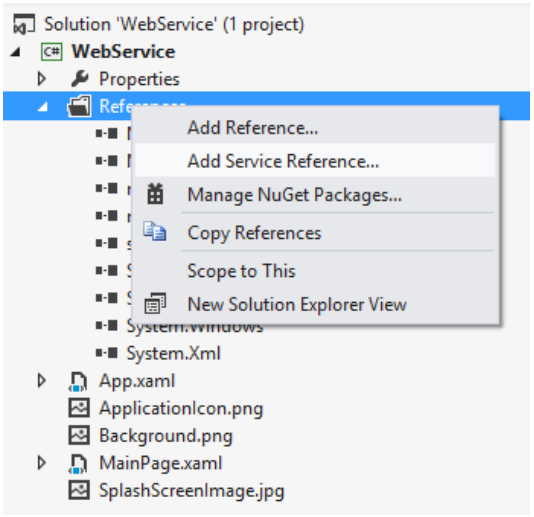
- Dále přidáme komponentu RadioButton pro přepínání mezi typy konverzí.

```
<!--ContentPanel - place additional content here-->
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
        <StackPanel>
                <TextBox x:Name="ZadanaTeplota" />
                <RadioButton x:Name="CtF" GroupName="Temp"
                             Content="Celsium -> Fahrenheit"
                             IsChecked="true" />
                <RadioButton x:Name="FtC" GroupName="Temp"
                             Content="Fahrenheit -> Celsium" />
        </StackPanel>
</Grid>
```

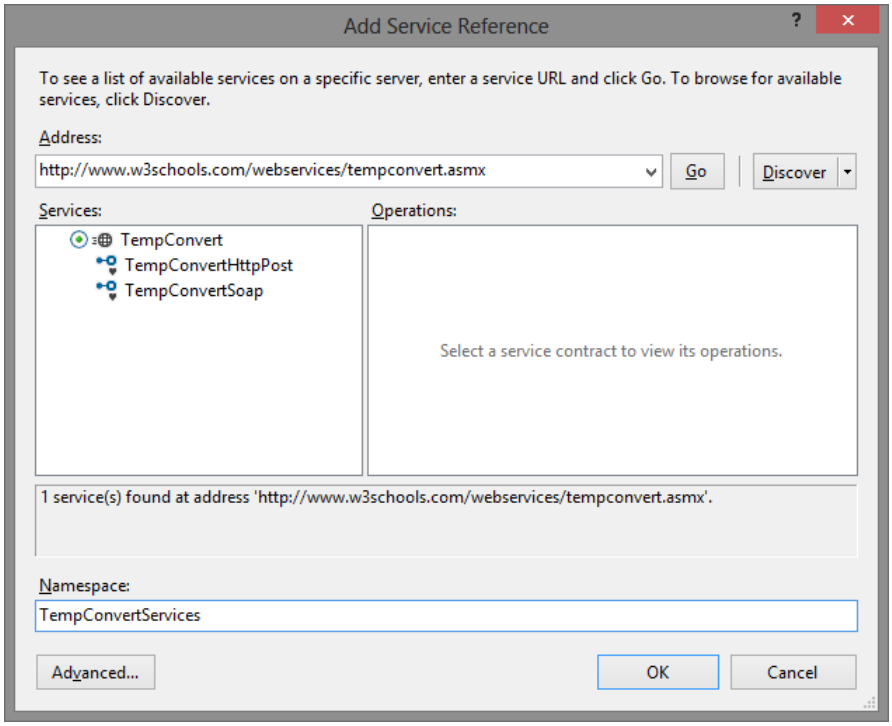
- V neposlední řadě přidáme komponentu TextBlock pro zobrazení konverze a také tlačítko, které tuto konverzi spouští.

```
<!--ContentPanel - place additional content here-->
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
        <StackPanel>
                <TextBox x:Name="ZadanaTeplota" />
                <RadioButton x:Name="CtF" GroupName="Temp"
                             Content="Celsium -> Fahrenheit"
                             IsChecked="true" />
                <RadioButton x:Name="FtC" GroupName="Temp"
                             Content="Fahrenheit -> Celsium" />
                <Button Content="konvertuj" />
                <TextBlock x:Name="VyslednaKonverze" FontSize="48"  />
        </StackPanel>
</Grid>
```

- Máme připravené uživatelské rozraní a nyní je nutné přidat referenci na webovou službu a logiku, která vybranou konverzi provede.
- Pro připojení k webovým službám použijeme položku *Add Service Reference* (Přidat referenci na službu) z kontextové nabídky projektu.



- Následně se nám otevře dialogové okno, kde zadáme adresu k webové službě a klikneme na tlačítko Go. Pro demonstraci příkladu jsme si vybrali webovou službu mající adresu http://www.w3schools.com/webservices/tempconvert.asmx.
- Jakmile je webová služba nalezena, zobrazí se název služby v podokně Services (Služby). Název služby, který nás zajímá, má název TempConvert.
- Poté pojmenujeme vhodně jmenný prostor (Namespace) pro službu, například TempConvertServices a potvrdíme tlačítkem OK.



- Po přidání takové webové služby do projektu se nám automaticky vygeneruje potřebný kód, který je nutný pro korektní komunikaci se službou.
- Třída, která je hlavním bodem pro komunikaci s webovou službou, obsahuje většinou ve svém názvu postfix Client. Pomoci této třídy můžeme přistupovat k webové službě. Všechna volání do webové služby opět probíhají asynchronně.
- V rámci tlačítka přidáme událost Click a necháme si vygenerovat obslužný kód.

```
<Button Content="konvertuj" Click="Button_Click" />
```

- V rámci obsluhy události přidáme kód pro konverzi. Vytvoříme si novou instanci třídy TempConvertSoapClient. Tato třída obsahuje metodu pro konvertování stupně Celsia na odpovídající stupně Fahrenheita a naopak, metody mají název CelsiusToFahrenheitAsync() a FahrenheitToCelsiusAsync(). Všimněte si opět postfixu Async. Volání metody služby se provede asynchronně, tedy je nutné obsloužit příslušné události. Pokud volání webové službu je úspěšné, má vlastnost Error hodnotu Null. Výsledek lze poté vyčíst pomoci vlastnosti Result, která je také vrácena prostřednictvím objektu CelsiusToFahrenheitCompletedEventArgs.

```
private void Button_Click(object sender, RoutedEventArgs e)
{
  // Nový klient, který je hlavním bodem při práci s webovou službou
  var tempConvertorClient = new TempConvertSoapClient();

  // Zadaná teplota ze vstupu
  string teplotaKeKonverzi = ZadanaTeplota.Text;

  // Je-li vybraná konverze z Celsia do Fahrenheita
  if ((bool)CtF.IsChecked) {

    // zavolá se asynchronně metoda pro konverze z Celsia do Fahrenheita
    tempConvertorClient.CelsiusToFahrenheitAsync(teplotaKeKonverzi);
    // Po dokončení asynchronního volání se zavolá událost Completed
    tempConvertorClient.CelsiusToFahrenheitCompleted += (s, a) =>
    {
    // Pokud prroběhlo vše v pořádku, tak nastavíme text na výsledek
       if (a.Error == null)
       {
           VyslednaKonverze.Text = a.Result;
       }
       else
       {
          // V případě chyby, alespoň informujeme uživatele
          VyslednaKonverze.Text = "Pri konverzi doslo k chybe";
       }
    };
 }
 else
 {
    tempConvertorClient.FahrenheitToCelsiusAsync(teplotaKeKonverzi);

    tempConvertorClient.FahrenheitToCelsiusCompleted += (s, a) =>
    {
        if (a.Error == null)
        {
            VyslednaKonverze.Text = a.Result;
        }
        else
        {
          VyslednaKonverze.Text = "Pri konverzi doslo k chybe";
        }
     };
  }
}
```

- V rámci konfiguračního souboru ServiceReference.ClientConfig webové služby můžeme zjistit adresu, kde se webová služba nachází. Adresa je definována v atributu address v elementu endpoint.

```
<configuration>
    <system.serviceModel>
        <bindings>
            <basicHttpBinding>
                <binding name="TempConvertSoap" maxBufferSize="2147483647"
                         maxReceivedMessageSize="2147483647">
                    <security mode="None" />
                </binding>
            </basicHttpBinding>
        </bindings>
      <client>
      <endpoint address="http://www.w3schools.com/webservices/tempconvert.asmx"
          binding="basicHttpBinding" bindingConfiguration="TempConvertSoap"
          contract="TempConvertServices.TempConvertSoap" name="TempConvertSoap" />
      </client>
    </system.serviceModel>
</configuration>
```

Výsledná aplikace pak vypadá následovně:

Obrázek 12 Výsledná aplikace ukázkové příkladu pro konverzi teploty

8. Doprovodný materiál

8.1. Příručka

Součástí práce je i rozsáhlejší příručka, která se věnuje detailnějšímu popisu a přidává navíc ukázky kódu. Také se zabývá pokročilejšími tématy, jako jsou například notifikace a jejich užití v praxi. Příručka obsahuje tipy a kontrolní otázky ke každé kapitole pro ověření znalostí. Tato příručka je dodávána s prací jako příloha.

8.2. Webová prezentace

Kromě samostatné příručky je dostupná i její webová verze, která se nachází na adrese http://www.winphone.cz. Členění sekcí a navigace je dodržována vzhledem k tištěné podobě. V rámci webové prezentace je také možnost si tuto příručku stáhnout.

9. Závěr

Cílem této práce bylo seznámit čtenáře s vývojem pro operační systém Windows Phone. Bylo zmíněno, co je potřeba pro efektivní vývoj aplikací, jaké komponenty lze využít a jaké specifické funkce systém nabízí.

Operační systém nabízí rychlou adaptaci pro vývojáře, kteří již v minulosti pracovali s nějakou technologii založenou na jazyce XAML. Pro tyto vývojáře, a ne jenom pro ně, může být platforma Windows Phone poměrně atraktivní.

Jak jsem již uvedl, vývoj pro Windows Phone je poměrně snadný, což lze také odpozorovat na množství aplikací, které denně přibývají do Marketplace. S tím samozřejmě souvisí i silná konkurence, která se ale příliš neliší od těch na jiných platformách.

Co může negativně ovlivňovat zájem o tuto platformu, je bouřlivý vývoj, který ji doprovází a také penetrace přístrojů, která jen pozvolna roste a je celkově za očekáváním. V současné době je protlačována nejnovější verze systému na úkor sedmičkové řady. V aktuální verzi systému už není bohužel možné psát aplikace pomoci Silverlightu a ani XNA. Nicméně filozofie vývoje aplikací založená na značkovacím jazyku XAML je velice podobná a tedy pevně doufám, že i přes tuto skutečnost, bude tato práce pro čtenáře přínosem v dalším rozvoji svých znalostí v oblasti vývoje mobilních aplikací.

Literatura

[1] Windows Phone Update History. In: *Windows Phone* [online]. [b.r.] [cit. 1.5.2013]. Dostupné z: http://www.windowsphone.com/en-us/how-to/wp7/basics/update-history

[2] Design Language of Windows Phone. In: *Microsoft .schools.tutorials* [online]. [b.r.] [cit. 1.5.2013]. Dostupné z: http://www.microsoft.com/design/toolbox/tutorials/windows-phone-7/metro/

[3] Hardware Specifications for Windows Phone. In: *MSDN Library* [online]. [b.r.] [cit. 1.10.2012]. Dostupné z: http://msdn.microsoft.com/en-us/library/ff637514(v=vs.92).aspx

[4] Hardware Specifications for Windows Phone. In: *MSDN Library* [online]. [b.r.] [cit. 1.10.2012]. Dostupné z: http://msdn.microsoft.com/en-us/library/ff637514(v=vs.92).aspx

[5] Application Platform Overview for Windows Phone. In: *MSDN Library* [online]. [b.r.] [cit. 1.10.2012]. Dostupné z: http://msdn.microsoft.com/en-us/library/ff402531%28VS.92%29.aspx

[6] App Activation and Deactivation for Windows Phone. In: *MSDN Library* [online]. [b.r.] [cit. 1.5.2013]. Dostupné z: http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff817008(v=vs.105).aspx

[7] XAML Overview. In: *MSDN Library* [online]. [b.r.] [cit. 1.5.2013]. Dostupné z: http://msdn.microsoft.com/en-us/library/cc189036(v=vs.95).aspx

[8] Microsoft Unveils Windows Phone 7 Series. In: *Microsoft News Center* [online]. 15. 10. 2012 [cit. 1.5.2013]. Dostupné z: https://www.microsoft.com/en-us/news/press/2010/feb10/02-15mwc10pr.aspx

[9] Tiles Overview for Windows Phone. In: *MSDN Library* [online]. [b.r.] [cit. 1.10.2012]. Dostupné z: http://msdn.microsoft.com/en-us/library/hh202948(v=VS.92).aspx

[10] Wildermuth, Shawn. *Essential Windows Phone 7.5: Application Development with Silverlight*. USA: Addison-Wesley, 2012. ISBN 978-0-321-75213-0

Seznam obrázků

Obrázek 1 Logo operačního systému Windows Phone 9
Obrázek 2 Ukázka Metro designu implementovaného v operačním systému Windows 8 . 11
Obrázek 3 Architektura WP7 [5] 13
Obrázek 4 Znázorněné stavy a události v životním cyklu aplikace 15
Obrázek 5 Hlavní stránka vývojového prostředí Visual Studio 2010 19
Obrázek 6 Výběr šablony pro vývoj Windows Phone aplikací ve vývojovém prostředí Visual Studio 2010 20
Obrázek 7 Popis vývojové prostředí Visual Studio 2010 21
Obrázek 8 Defaultní vzhled komponenty Panorama 29
Obrázek 9 Defaultní vzhled komponenty Pivot 30
Obrázek 10 Živá dlaždice a její struktura 31
Obrázek 11 Ukázka dlaždice 32
Obrázek 12 Výsledná aplikace ukázkové příkladu pro konverzi teploty 42

Přílohy

1. **Příručka o vývoji pro Windows Phone 7**, umístěna v elektronické podobě na CD a v tištěné podobě
2. **CD,** obsahuje elektronickou verzi bakalářské práce